CLAIMS

1. A compiler program for converting a source program into an object program and causing a computer to function as:

an address saving program generating means for generating an address saving program for saving a data memory area address used by a calling program module included in the source program,

an address setting program generating means for generating an address setting program for setting a data memory area address used by an other program module to be called by the calling program module,

a transferring program generating means for generating a transferring program for the transfer from a first subprogram included in the calling program module to a second subprogram included in the other program module,

an address resetting program generating means for generating an address resetting program for reading and resetting the saved data memory area address for the calling program module after the return from the second subprogram as a transfer end to the first subprogram, and

an accessing program generating means for generating an accessing program for accessing a data memory area for the other program module using a relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein.

2. A compiler program according to claim 1, wherein:

the computer is caused to further function as a discriminating means for discriminating whether or not to shorten a process based on a description of the other program module including the second subprogram to be called from the first subprogram included in the calling program module included in the source program,

if the discriminating means discriminates the process not to be shortened,

the address saving program generating means generates the address saving program for saving the data memory area address used by the calling program module,

the address setting program generating means generates the address setting program for setting the data memory area address used by the other program module to be called by the calling program module,

the transferring program generating means generates the transferring program for the transfer from the first subprogram included in the calling program module to the second subprogram included in the other program module,

the address resetting program generating means generates the address resetting program for reading and resetting the saved data memory area address for the calling program module after the return from the second

subprogram as the transfer end to the first subprogram, and

the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein, and

if the discriminating means discriminates the process to be shortened,

the transferring program generating means generates the transferring program for the transfer from the first subprogram included in the calling program module to the second subprogram included in the other program module, and

the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the data memory area address for the calling program module when the other program module accesses the data memory area therefor included therein.

3. A compiler program according to claim 1, wherein, after the second subprogram included in the other program module is called from the first subprogram included in the calling

program module included in the source program;

the address setting program generating means generates an address setting program for reading the data memory area address for the other program module from data memory area address tables for the respective program modules saved in executing units of the calling program module and setting this data memory area address, and

the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein.

4. A compiler program according to claim 3, wherein the computer is caused to further function as a discriminating means for discriminating whether or not to shorten a process based on a description of the other program module including the second subprogram to be called from the first subprogram included in the calling program module included in the source program,

if the discriminating means discriminates the process not to be shortened, after the second subprogram included in the other program module is called from the first subprogram included in the calling program module included in the source

program;

the address setting program generating means generates the address setting program for reading the data memory area address for the other program module from the data memory area address tables for the respective program modules saved in executing units of the calling program module and setting this data memory area address, and

the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein, and

if the discriminating means discriminates the process to be shortened,

the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the data memory area address for the calling program module when the other program module accesses the data memory area therefor included therein.

5. A compiler program according to claim 2 or 4, wherein the description of the other program module including

the second subprogram to be called from the first subprogram included in the calling program module included in the source program is a declaration described for each second subprogram.

6. A compiler program according to any one of claims 1 to 5, wherein the source program including the other program module including the second subprogram to be called by the calling program module includes a description for specifying which of an execution common data memory area for the other program module external variables included in the other program module commonly access upon a plurality of executions of the calling program module and an execution intrinsic data memory areas for the other program module the external variables included in the other program module commonly access every time the calling program module is executed is to be used.

7. A compiler program according to claim 6, wherein the description for specifying which of the execution common data memory area for the other program module and the execution intrinsic data memory area for the other program module is to be used is a declaration described for each external variable included in the other program module.

8. A compiler program according to claim 6, wherein:

the address saving program generating means generates an

address saving program for saving an execution common data memory area address for the calling program module and an execution intrinsic data memory area address for the calling program module,

the address setting program generating means generates an address setting program for setting the execution common data memory area address for the other program module to be called by the calling program module and setting the execution intrinsic data memory area address for the other program module to be called by the calling program module,

the transferring program generating means generates the transferring program for the transfer from the first subprogram to the second subprogram,

the address resetting program generating means generates an address resetting program for reading and resetting the saved execution intrinsic data memory area address and execution common data memory area address for the calling program module after the return from the second subprogram as the transfer end to the first subprogram, and

the accessing program generating means generates an accessing program for accessing the execution intrinsic data memory area for the other program using a relative address from the set execution intrinsic data memory area address for the other program module when the other program module accesses the execution intrinsic data memory area address therefor included

therein and accessing the execution common data memory area for the other program using a relative address from the set execution common data memory area address for the other program module when the other program module accesses the execution intrinsic data memory area address therefor included therein.

9. A compiler program according to claim 6, wherein, after the other program module is called from the calling program module included in the source program:

the address setting program generating means generates an address setting program for reading the execution intrinsic data memory area address for the other program module and the execution common data memory area address for the other program module commonly accessed by a plurality of executions of the calling program module from the data memory area address tables for the respective program modules saved in executing units of the calling program module and setting these data memory area addresses, and

the accessing program generating means generates an accessing program for accessing the execution intrinsic data memory area for the other program using a relative address from the set execution intrinsic data memory area address for the other program module when the other program module accesses the execution intrinsic data memory area address therefor included therein and accessing the execution common data memory area for

the other program using a relative address from the set execution common data memory area address for the other program module when the other program module accesses the execution intrinsic data memory area address therefor included therein.

10. A compiler program according to claim 1, wherein the address saving program generating means generates an address saving program for saving the execution common data memory area address and the execution intrinsic data memory area address for the calling program module in a stack memory prior to data necessary for the transfer from the first subprogram to the second subprogram.

11. A compiler program according to any one of claims 1 to 10, wherein the calling program module included in the source program and the other program module including the second subprogram to be called from the first subprogram included in the calling program module are the same program module.

12. A compiler program according to claim 1, wherein the calling program module includes a code area and a data area, the other program module includes a code area and a data area, and the computer is caused to further function as:

an identification-number obtaining program generating means for generating an identification-number obtaining program

for obtaining an identification number for identifying the other program module upon the receipt of a call command to the other program module from the calling program module,

a leading-address obtaining program generating means for generating a leading-address obtaining program for obtaining the leading address of the data area of the other program module using the obtained identification number as an index from a table relating an identification number of the calling program module to the leading address of the data area of the calling program module and relating the identification number of the other program module to the leading address of the data area of the other program module, and

a leading-address switching program generating means for generating a leading-address switching program for switching the obtained leading address of the data area of the other program module and the leading address of the data area of the calling program module.

13. A compiler program according to claim 1, wherein the calling program module includes a code area and a data area, the other program module includes a code area and a data area, and the computer is caused to further function as:

an identification-number designating program generating means for generating an identification-number generating program for designating an identification number for identifying the

calling program module and an identification number for identifying the other program module,

a table preparing program generating means for generating a table preparing program for preparing a table relating the designated identification number of the calling program module to the leading address of the data area of the calling program module and relating the designated identification number of the other program module to the leading address of the data area of the other program module,

an identification-number obtaining program generating means for generating an identification-number obtaining program for obtaining the identification number of the other program module upon the receipt of a call command to the other program module from the calling program module,

a leading-address obtaining program generating means for generating a leading-address obtaining program for obtaining the leading address of the data area of the other program module from the prepared table using the obtained identification number as an index, and

a leading-address switching program generating means for generating a leading-address switching program for switching the obtained leading address of the data area of the other program module and the leading address of the data area of the calling program module.

14. A compiler program according to claim 1, wherein the computer is caused to further function as an address latching program generating means for generating an address latching program for latching an address of a variable or function defined outside the other program module, and the accessing program generating means generates an accessing program for accessing a variable defined inside the other program module using a relative address from the obtained leading address, accesses a function defined inside the other program module using a relative address from a program counter, and indirectly accessing the variable or function defined outside the other program module via the latched address after accessing the latched address using a relative address from the obtained leading address.

15. A compiler program according to claim 1, wherein the computer is caused to further function as:

an ending program generating means for generating an ending program for ending an application comprised of at least one module and being presently executed,

a compaction program generating means for generating a compaction program for compacting a memory after the application is ended, and

a resuming program generating means for generating a resuming program for resuming the application from the beginning

after the memory is compacted.

16. A compiler program according to claim 1, wherein the computer is caused to further function as:

an ending program generating means for generating an ending program for ending an application comprised of at least one module and being presently executed,

a saving program generating means for generating a saving program for saving information independent of address values of the application until the application was ended,

a compaction program generating means for generating a compaction program for compacting a memory, and

an executing program generating means for generating an executing program for reading the saved information and executing the application up to a state where the application was ended based on the read information after the memory is compacted.

17. A compiler program according to claim 1, wherein an identifier indicating whether or not to save an address value is provided in a memory, and the computer is caused to further function as:

a memory-state saving program generating means for generating a memory state saving program for saving the state of the memory before the compaction,

a compaction program generating means for generating a compaction program for compacting the memory, and

a relocating program generating means for generating a relocating program for relocating the address value based on the identifier for an entry having the address value set by referring to the saved state of the memory before the compaction, after the memory is compacted.

18. A compiler program according to claim 1, wherein an address-value memory for saving address values and a non-address-value memory for saving values other than address values are provided, and the computer is caused to further function as:

a memory-state saving program generating means for generating a memory-state saving program for saving the states of the address-value memory and the non-address-value memory before the compaction,

a compaction program generating means for generating a compaction program for compacting the address-value memory and the non-address-value,

a relocating program generating means for generating a relocating program for relocating the address values of only the address-value memory by referring to the saved state of the address-value memory before the compaction, after the address-value memory and the non-address-value memory are compacted.

19. A compiler program according to claim 1, wherein identification numbers for identifying the modules and identifiers indicating whether to set addresses of data areas of the modules or to set addresses of code areas of the modules are provided in a memory, and the computer is caused to further function as:

a setting program generating means for generating a setting program for setting the identification number and the identifier upon substituting an address value for a pointer,

a compaction program generating means for generating a compaction program for compacting the memory, and

a recalculating program generating means for generating a recalculating program for recalculating the address value based on the identification number and the identifier after the memory is compacted.

20. A compiler program according to claim 1, wherein identification numbers for identifying the modules and identifiers indicating whether to set addresses of data areas of the modules or to set addresses of code areas of the modules are provided in a memory, and the computer is caused to further function as:

an offset-value setting program generating means for generating an offset-value setting program for setting the identification number and the identifier upon substituting an

address value for a pointer and setting an offset value from the leading address of the data area or the leading address of the code area, and

an actual-address converting program generating means for generating an actual-address converting program for the conversion into an actual address by adding the leading address of the data area or the leading address of the code area to the offset value upon using the pointer.

21. A compiler program according to claim 1, wherein the computer is caused to further function as:

a compaction program generating means for generating a compaction program for compacting each code area of the modules or each data area of the modules,

a searching program generating means for generating a searching program for searching whether or not there is any handle indicating the code area or the data area shifted by the compaction, and

a handle relocating program generating means for generating a handle relocating program for relocating the handle if the handle indicating the code area or the data area is found,

wherein the compaction program generating means generates the compaction program for compacting the code areas or the data areas after all the handles are relocated.

22. A computer-readable storage medium storing a compiler program for converting a source program into an object program and causing a computer to function as:

an address saving program generating means for generating an address saving program for saving a data memory area address used by a calling program module included in the source program,

an address setting program generating means for generating an address setting program for setting a data memory area address used by an other program module to be called by the calling program module,

a transferring program generating means for generating a transferring program for the transfer from a first subprogram included in the calling program module to a second subprogram included in the other program module,

an address resetting program generating means for generating an address resetting program for reading and resetting the saved data memory area address for the calling program module after the return from the second subprogram as a transfer end to the first subprogram, and

an accessing program generating means for generating an accessing program for accessing a data memory area for the other program module using a relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein.

23. A compiling method for converting a source program into an object program, comprising:

a step of causing a computer to generate an address saving program for saving a data memory area address used by a ca program module included in the source program,

a step of causing the computer to generate an address setting program for setting a data memory area address used by an other program module to be called by the calling program module,

a step of causing the computer to generate a transferring program for the transfer from a first subprogram included in the calling program module to a second subprogram included in the other program module,

a step of causing the computer to generate an address resetting program for reading and resetting the saved data memory area address for the calling program module after the return from the second subprogram as a transfer end to the first subprogram, and

a step of causing the computer to generate an accessing program for accessing a data memory area for the other program module using a relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein.

24. A compiling unit for converting a source program into an object program, comprising:

an address saving program generating means for generating an address saving program for saving a data memory area address used by a calling program module included in the source program,

an address setting program generating means for generating an address setting program for setting a data memory area address used by an other program module to be called by the calling program module,

a transferring program generating means for generating a transferring program for the transfer from a first subprogram included in the calling program module to a second subprogram included in the other program module,

an address resetting program generating means for generating an address resetting program for reading and resetting the saved data memory area address for the calling program module after the return from the second subprogram as a transfer end to the first subprogram, and

an accessing program generating means for generating an accessing program for accessing a data memory area for the other program module using a relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein.

25. An object program generated by converting a source

program and causing a computer to functions as:

an address saving means for saving a data memory area address used by a calling program module included in the source program,

an address setting means for setting a data memory area address used by an other program module to be called by the calling program module,

a transferring means for executing the transfer from a first subprogram included in the calling program module to a second subprogram included in the other program module,

an address resetting means for reading and resetting the saved data memory area address for the calling program module after the return from the second subprogram as a transfer end to the first subprogram, and

an accessing means for accessing a data memory area for the other program module using a relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein.

26. A computer-readable storage medium storing an object program generated by converting a source program and causing a computer to functions as:

an address saving means for saving a data memory area address used by a calling program module included in the source

program,

an address setting means for setting a data memory area address used by an other program module to be called by the calling program module,

a transferring means for executing the transfer from a first subprogram included in the calling program module to a second subprogram included in the other program module,

an address resetting means for reading and resetting the saved data memory area address for the calling program module after the return from the second subprogram as a transfer end to the first subprogram, and

an accessing means for accessing a data memory area for the other program module using a relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein.

27. An object program executing method for executing an object program generated by converting a source program, comprising:

an address saving step of causing a computer to save a data memory area address used by a calling program module included in the source program,

an address setting step of causing the computer to set a data memory area address used by an other program module to be

called by the calling program module,

a transferring step of causing the computer to execute the transfer from a first subprogram included in the calling program module to a second subprogram included in the other program module,

an address resetting step of causing the computer to read and reset the saved data memory area address for the calling program module after the return from the second subprogram as a transfer end to the first subprogram, and

an accessing step of causing the computer to access a data memory area for the other program module using a relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein.

28. An object program executing unit for executing an object program generated by converting a source program, comprising:

an address saving means for saving a data memory area address used by a calling program module included in the source program,

an address setting means for setting a data memory area address used by an other program module to be called by the calling program module,

a transferring means for executing the transfer from a

first subprogram included in the calling program module to a second subprogram included in the other program module,

an address resetting means for reading and resetting the saved data memory area address for the calling program module after the return from the second subprogram as a transfer end to the first subprogram, and

an accessing means for accessing a data memory area for the other program module using a relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein.

29. An object program executing unit according to claim 28, wherein the calling program module includes a code area and a data area, the other program module includes a code area and a data area, and the object program executing unit further comprises:

an identification-number designating means for designating an identification number for identifying the calling program module and an identification number for identifying the other program module,

a table preparing means for preparing a table relating the identification number of the calling program module designated by the identification-number designating means to the leading address of the data area of the calling program module

and relating the identification number of the other program module designated by the identification-number designating means to the leading address of the data area of the other program module,

an identification-number obtaining means for obtaining the identification number of the other program module upon the receipt of a call command to the other program module from the calling program module,

a leading-address obtaining means for obtaining the leading address of the data area of the other program module from the table prepared by the table preparing means using the identification number obtained by the identification-number obtaining means as an index, and

a leading-address switching means for switching the leading address of the data area of the other program module obtained by the leading-address obtaining means and the leading address of the data area of the calling program module.

30. An object program executing unit according to claim 28, wherein the calling program module includes a code area and a data area, the other program module includes a code area and a data area, and the object program executing unit further comprises:

an identification-number designating means for designating an identification number for identifying the calling

program module and an identification number for identifying the other program module, and

a table preparing means for preparing a table relating the identification number of the calling program module designated by the identification-number designating means to the leading address of the data area of the calling program module and relating the identification number of the other program module designated by the identification-number designating means to the leading address of the data area of the other program module.